

1 KEKER & VAN NEST LLP
2 ROBERT A. VAN NEST - # 84065
3 rvannest@kvn.com
4 CHRISTA M. ANDERSON - # 184325
5 canderson@kvn.com
6 MICHAEL S. KWUN - # 198945
7 mkwun@kvn.com
8 633 Battery Street
9 San Francisco, CA 94111-1809
10 Tel: 415.391.5400
11 Fax: 415.397.7188

KING & SPALDING LLP
DONALD F. ZIMMER, JR. - #112279
fzimmer@kslaw.com
CHERYL A. SABNIS - #224323
csabnis@kslaw.com
101 Second Street, Suite 2300
San Francisco, CA 94105
Tel: 415.318.1200
Fax: 415.318.1300

7 KING & SPALDING LLP
8 SCOTT T. WEINGAERTNER
9 (Pro Hac Vice)
10 sweingaertner@kslaw.com
11 ROBERT F. PERRY
12 rperry@kslaw.com
13 BRUCE W. BABER (Pro Hac Vice)
14 1185 Avenue of the Americas
15 New York, NY 10036
16 Tel: 212.556.2100
17 Fax: 212.556.2222

IAN C. BALLON - #141819
ballon@gtlaw.com
HEATHER MEEKER - #172148
meekerh@gtlaw.com
GREENBERG TRAURIG, LLP
1900 University Avenue
East Palo Alto, CA 94303
Tel: 650.328.8500
Fax: 650.328.8508

13 Attorneys for Defendant
14 GOOGLE INC.

15 UNITED STATES DISTRICT COURT
16 NORTHERN DISTRICT OF CALIFORNIA
17 SAN FRANCISCO DIVISION

18 ORACLE AMERICA, INC.,
19 Plaintiff,
20 v.
21 GOOGLE INC.,
22 Defendant.

Case No. 3:10-cv-03561 WHA

**GOOGLE'S MAY 10, 2012 COPYRIGHT
LIABILITY TRIAL BRIEF**

Dept.: Courtroom 8, 19th Floor
Judge: Hon. William Alsup

TABLE OF CONTENTS

	<u>Page</u>
I. Google requests that the Court hold that the structure, sequence and organization (“SSO”) of the 37 API packages is not copyrightable.....	1
II. Google’s responses to the Court’s questions	1
A. Question 1: The vocabulary and grammar of a computer language—as distinct from programs written in the language—are not copyrightable because the Copyright Act protects expression, not vocabulary.	1
1. Copyright does not protect vocabulary, and that lack of protection extends to the SSO of vocabulary.....	1
2. The ECJ’s decision in <i>SAS Inst. Inc. v. World Programming Ltd</i> supports the conclusion that the vocabulary of a programming language cannot be copyrighted.....	4
B. Question 2: If each API method is treated as a program, then Google did not copy anything more than the name and declaration of that program, which is not copyrightable.	5
C. Question 3: The form of the fully qualified names of the methods in the 37 API packages is “package.class.method,” where the package names start with “java.” or “javax.” and this form is required by the syntax of the Java language.	6
D. Question 4: Google could have come up with at least some different names and SSO yet still provided similar functionality in Android, but this would not have been consistent with industry custom and developer demand.	6
1. Many of the elements of the 37 API packages are <i>required</i> by the Java language.	7
2. Industry custom and developer demand require implementation of the 37 API packages.....	8
3. When developing new APIs in the Java language, it is standard practice to build upon the standard J2SE APIs.	9
4. The Court should disregard Mr. Ellison’s testimony about Spring.	9
E. Question 5: The input-output scheme of a method is not copyrightable.	10
F. Question 6: The “core” packages in 1996 included <i>at least</i> java.lang, java.io and java.util, and today include at least 34 of the accused packages.....	10
G. Question 7: The Java language requires more than just the java.lang, java.io and java.util packages.	11
H. Question 8: All 37 accused API packages should be deemed “core” packages.	12

1	I.	Question 9: There are cross-method, cross-class interdependencies <i>at the</i>	
2		<i>implementation level</i> in J2SE, and those implementation level	
3		interdependencies are not always duplicated in the Android	
		implementations.....	12
4	J.	Question 10: At the name/declaration level, the only interdependencies in	
5		the Java language come from (1) the package/class/member organizational	
		scheme; (2) inheritance via subclassing and subinterfacing; and	
		(3) interface implementation.....	13
6	K.	Question 11: The Seventh Circuit’s decision in <i>American Dental Ass’n v.</i>	
7		<i>Delta Dental Plans Ass’n</i> , was either wrongly decided, or unclear about the	
		scope of its holding.	13
8	L.	Question 12: <i>CDN, Inc. v. Kapes</i> was not about a method of operation or	
9		system.	15
10	M.	Question 13: Android is compatible with the 37 API packages at issue	
11		because code written to use those APIs will compile and work properly on	
		an Android device.	16
12	N.	Questions 14-15: In the Java language, “inheritance” is a concept	
13		applicable to classes, not packages; by virtue of the Java language	
		specification, a subclass inherits fields, methods, and other members from	
		its superclass.	17
14	O.	Question 16: Copyright protection does not extend to names, including	
15		fully qualified names, nor does it extend to input-output (argument-return)	
16		designations, exception types, subclass inheritances, or interface	
		relationships.	18
17	III.	The Court should hold that the SSO of the 37 API packages is not copyrightable.	18

TABLE OF AUTHORITIES

Page(s)**Federal Cases**

<i>Allen v. Academic Games League of Am., Inc.</i> 89 F.3d 614 (9th Cir. 1996)	6, 10
<i>ATC Distrib. Grp., Inc. v. Whatever It Takes Trans. & Parts, Inc.</i> 402 F.3d 700 (6th Cir. 2005)	5, 14
<i>Baker v. Selden</i> 101 U.S. 99 (1879).....	1, 6
<i>Computer Assocs. Int'l, Inc. v. Altai, Inc.</i> 982 F.2d 693 (2d Cir. 1992).....	8
<i>Engineering Dynamics, Inc. v. Structural Software, Inc.</i> 46 F.3d 408 (5th Cir. 1995)	10
<i>Feist Pubs., Inc. v. Rural Tele. Serv. Co.</i> 499 U.S. 340 (1991).....	3, 4, 15
<i>Gates Rubber Co. v. Bando Chem. Indus., Ltd</i> 9 F.3d 823 (10th Cir. 1993)	8
<i>Herbert Rosenthal Jewelry Corp. v. Kalpakian</i> 446 F.2d 738 (9th Cir. 1971)	6
<i>Lotus Dev. Corp. v. Borland Int'l, Inc.</i> 49 F.3d 807 (1st Cir. 1995), <i>aff'd by an equally divided court</i> , 516 U.S. 233 (1996)	6
<i>Sega Enters. Ltd v. Accolade, Inc.</i> 977 F.2d 1510 (9th Cir. 1992)	1, 6, 8, 10, 16
<i>Seng-Tiong Ho v. Taflove</i> 648 F.3d 489 (7th Cir. 2011)	13
<i>Southco, Inc. v. Kanebridge Corp.</i> 258 F.3d 148 (3d Cir. 2001).....	13
<i>Swirsky v. Carey</i> 376 F.3d 841 (9th Cir. 2004)	8

Other Cases

<i>SAS Inst. Inc. v. World Programming Ltd</i> Case C-406/10 (ECJ May 2, 2012)	4, 5
---	------

Federal Statutes

17 U.S.C. § 102.....	1
----------------------	---

1 17 U.S.C. § 102(b) *passim*

2 **Federal Rules**

3 Fed. R. Evid. 201(b)(2) 9

4 **Federal Regulations**

5 37 C.F.R § 202.1(a) 2, 5

I. Google requests that the Court hold that the structure, sequence and organization (“SSO”) of the 37 API packages is not copyrightable.

As directed by the Court, Google provides responses to the sixteen questions the Court has posed. *See* Dkts. 1057, 1062, 1088. Google also offers further argument in support of the conclusion that the SSO of the 37 API packages is not copyrightable.

For the reasons expressed below and in Google’s prior filings, and based on the trial record, the SSO of the 37 API packages is not copyrightable. This conclusion of law is independently supported by (1) Oracle’s concession that the Java language is free and open for anyone to use; (2) the “system” and “method of operation” exclusions in section 102(b) of the Copyright Act; (3) the “functional requirements for compatibility” interpretation of section 102(b) adopted by the Ninth Circuit in *Sega Enters. Ltd v. Accolade, Inc.*, 977 F.2d 1510, 1522 (9th Cir. 1992); (4) constraints on the SSO imposed by the requirements of the Java language and the merger doctrine; and (5) expectations of developers and industry and the *scenes a faire* doctrine.

II. Google’s responses to the Court’s questions:

A. Question 1: The vocabulary and grammar of a computer language—as distinct from programs written in the language—are not copyrightable because the Copyright Act protects expression, not vocabulary.

1. Copyright does not protect vocabulary, and that lack of protection extends to the SSO of vocabulary.

Well over a century ago, in a decision that is now codified at 17 U.S.C. § 102(b), the Supreme Court announced a fundamental principle dividing copyright and patent protection: “To give the author of the book an exclusive property in the art described therein, when no examination of its novelty has ever been officially made, would be a surprise and a fraud upon the public. That is the province of letters-patent, not copyright.” *Baker v. Selden*, 101 U.S. 99, 102 (1879). Throughout the first phase of this trial, Oracle has repeatedly attempted to circumvent *Baker*’s holding by claiming copyright protection in elements of its overall platform, through claims to the Java vocabulary and vague claims to that vocabulary’s “structure, sequence and organization.”

The Copyright Act protects expression, not systems for expression. *See* 17 U.S.C. § 102; *Baker*, 101 U.S. at 102. Copyright does not protect vocabulary—for example, it does not protect

1 names, words or short phrases. 37 C.F.R § 202.1(a); Copyright MSJ Order [Dkt. 433] at 7:25-
2 8:4. Nor does it help Oracle's position to argue that the particular structure, sequence and
3 organization of its vocabulary should somehow be protected, even if the individual elements of
4 that vocabulary are not protected.

5 First, the "structure" or "organization" of words in relation to other words is not
6 protectable. In any language, words are related to other words, and concepts are related to other
7 concepts. Without such interrelations, a vocabulary has no value. Some words are, by definition,
8 more specific versions of others (e.g., a "plane" is a particular type of "vehicle"), just as a
9 subclass in the Java language is a more specific version of its superclass. Indeed, this is precisely
10 the analogy Dr. Reinhold used in explaining the concept of subclass inheritance. RT 587:10-
11 588:11 (explaining how a "Vehicle" class could be the superclass for the "Car," "Train" and
12 "Plane" classes). Dr. Reinhold similarly explained interfaces, fields and methods by reference to
13 properties of real-world items that people describe with words every day. For example, cars have
14 engines that can be started, horns that the driver can "blow," and lights that can be turned on. RT
15 588:12-20 (explaining methods that the Car class could have). He explained interfaces by
16 analogy to real-world items, noting that there are things other than vehicles that have horns, and
17 thus it is possible to have an interface that reflects this fact. RT 590:5-17 (discussing
18 "ThingWithHorn" interface). The "interdependencies" that Oracle has relied upon so frequently
19 are, in the end, nothing special, and nothing protectable under copyright law. In any useful
20 vocabulary, there are similar relationships that can be drawn, as Dr. Reinhold's analogies
21 demonstrate. Thus, the placement of API elements in packages or classes, and the hierarchical
22 arrangement of those elements via subclassing and interface implementation, all in accordance
23 with the requirements of the Java language, does nothing to make them more copyrightable than
24 any words or short phrases in the English language.

25 Second, the "sequence" of the elements in the API does not amount to creative expression.
26 Nothing in the record suggests that Google copied the sequence in which the APIs are
27 implemented within the source code. For example, the methods in java.lang.Math are
28 implemented in entirely different orders in Android and J2SE. *Compare* TX 47.101 (Android

1 Gingerbread version of “Math.java”) *with* TX 623.101 (J2SE 5.0 version of “Math.java”). The
 2 first ten public methods declared in the Android version are:

3 double abs(double d)
 4 float abs(float f)
 5 int abs(int i)
 6 long abs(long l)
 7 double acos(double d)
 8 double asin(double d)
 9 double atan(double d)
 10 double atan2(double y, double x)
 11 double cbrt(double d)
 12 double ceil(double d)

13 *See* TX 47.101 at lines 61, 83, 100, 113, 133, 151, 171, 211, 230, 250. The first ten public
 14 methods declared in the J2SE version are very different:

15 double sin(double a)
 16 double cos(double a)
 17 double tan(double a)
 18 double asin(double a)
 19 double acos(double a)
 20 double atan(double a)
 21 double toRadians(double angdeg)
 22 double toDegrees(double angrad)
 23 double exp(double a)
 24 double log(double a)

25 *See* TX 623.101 at lines 103, 118, 135, 153, 169, 186, 200, 216, 236, 257. The methods are
 26 ordered *alphabetically* in the documentation for both Android and J2SE, but that “choice” is so
 27 unoriginal that it is not protectable. *See Feist Pubs., Inc. v. Rural Tele. Serv. Co.*, 499 U.S. 340,
 28 362 (1991) (alphabetical list of subscribers is “devoid of even the slightest trace of creativity”).

29 Third, even if the SSO of a vocabulary could theoretically be protected—and on this
 30 record, there is no evidence suggesting that *Oracle’s* SSO should be protected—the SSO would
 31 only be protected *as a whole*. “No matter how original the format, however, the facts themselves
 32 do not become original through association.” *Feist*, 499 U.S. at 349. Oracle has not argued,
 33 however, that Google copied its SSO *as a whole*. Instead, it argues that Google adopted *part of*
 34 that SSO, and integrated it with further API packages of Google’s own design. Indeed, that
 35 fact—the alleged “fragmentation” caused by Android—is the crux of Oracle’s claim of harm.
 36 That is, Oracle complains that Google *did not* adopt the SSO of Oracle’s 166 API packages, but
 37 instead forged forward with a *different* SSO of Google’s own design. Google was free to do so:

1 “Notwithstanding a valid copyright, a subsequent compiler remains free to use the facts contained
 2 in another’s publication to aid in preparing a competing work, so long as the competing work
 3 does not feature *the same* selection and arrangement.” *Feist*, 499 U.S. at 349 (emphasis added).

4 **2. The ECJ’s decision in *SAS Inst. Inc. v. World Programming Ltd***
 5 **supports the conclusion that the vocabulary of a programming**
 6 **language cannot be copyrighted.**

7 Interpreting the same broad legal principles that are at issue in the present case, the
 8 European Court of Justice recently held that computer programming languages are not
 9 copyrightable. *SAS Inst. Inc. v. World Programming Ltd*, Case C-406/10 (ECJ May 2, 2012)
 10 [Dkt. 1047-1].¹ One of the questions before the ECJ was whether a “Second Program” that
 11 “replicates the functions of [a] First Program” infringes the copyright in the First Program. *Id.*
 12 ¶ 28(1). More specifically, the ECJ was asked to consider the situation where the First Program
 13 “interprets and executes programs written by users of the First Program in a programming
 14 language devised by the author of the First Program and a syntax devised by the author of the
 15 First Program” and where the Second Program is “written so as to interpret and execute such
 16 application programs using the same keywords and the same syntax.” *Id.* ¶ 28(3). This is, in
 17 essence, the question in the present case. Developers use the methods, fields, constructors and
 18 initializers of the 37 API packages to write programs in the Java language, and Android has been
 19 written such that it can interpret and execute those programs to the extent they use methods,
 20 fields, constructors and initializers found in the 37 packages. RT 2171:24-2172:11 (Astrachan);
 21 RT 2292:25-2293:14 (Mitchell).

22 The ECJ concluded that “the programming language and the format of data files” are
 23 “elements of [the First Program] by means of which users exploit functions of that [First
 24 P]rogram.” Dkt. 1047-1 ¶ 42. The programming language is not “a form of expression of that
 25 program” for purposes of copyright law. *Id.* ¶ 39. Any other conclusion “would amount to
 26 making it possible to monopolise ideas, to the detriment of technological progress and industrial

27 ¹ The European Committee for Interoperable Systems—of which Oracle is a member, *see*
 28 <http://www.ecis.eu/about-ecis/>—lauded the decision. *See, e.g., Reuters, EU court limits*
copyright protection for software (May 2, 2012) (including quote from Thomas Vinje, a
 spokesperson for the ECIS, supporting the ECJ’s decision).

development.” *Id.* ¶ 40.²

While the ECJ was applying European law, the same principles apply here. Moreover, the ECJ’s conclusion was that the programming language cannot be protected by copyright law, because it is on the unprotectable idea side of the idea/expression dichotomy. It necessarily follows that no copyright protection prevents others from adopting *parts* of a programming language. Similarly, copyright law cannot prevent Google from adopting *parts* of the 166 J2SE API packages.

B. Question 2: If each API method is treated as a program, then Google did not copy anything more than the name and declaration of that program, which is not copyrightable.

Google used nothing more than the name and declaration of each API element. For example, Google used the same method declarations³—which necessarily means Google used the same names. A method declaration includes the method’s name, the parameters it accepts as input, and the type of thing that it returns. RT 785:25-787:8 (Bloch); *see generally* TX 984 (*The Java Language Specification*, Third Edition) at 209-37. A method declaration can also include modifiers such as “public” or “final.” *See* TX 984 at 214-20. In addition, a method declaration can use the term “throws” to indicate “exceptions” (errors) that the method can communicate. *See* TX 984 at 221-23. The method body—sometimes referred to at trial as the *implementation* of the method, *see, e.g.*, RT 790:20-23 (Bloch); RT 1566:23-1567:10 (Schmidt); RT 2186:3-12 (Astrachan)—is “a block of code that implements the method” TX 984 at 223. Aside from the nine-line *rangeCheck* method, there is no dispute that Google’s implementing code is different from Oracle’s. RT 1309:8-1313:11 (Mitchell); RT 2182:13-2183:1 (Astrachan). The declarations are, in essence, the titles of the things they declare. Titles are not copyrightable. 37

² That said, if the defendant copied *implementing code* from the First Program, that could constitute infringement. *Id.* ¶ 43 (“it should be made clear that, if a third party were to *procure the part of the source code or the object code* relating to the programming language or to the format of data files used in a computer program, and if that party were to create, *with the aid of that code*, similar elements in its own computer program, that conduct would constitute partial reproduction”) (emphasis added).

³ As the Court has noted, the formal definition of a “method declaration” includes the “method body”—the implementation. TX 984 at 209-10. At trial, however, witnesses excluded the method body from their definition of “method declaration” and distinguished between the “declaration” and the “implementing code.”

1 C.F.R. § 202.1(a).

2 Moreover, the declarations simply state the functional characteristics of the API
3 elements—the names by which they can be invoked, the parameters they must be given as input,
4 and so on. RT 1773:25-1774:25 (Bornstein); RT 2106:13-2109:2 (Astrachan). In order to create
5 API implementations that are compatible with the J2SE APIs—that is, implementations that will
6 function in the same way when called by code written by developers—these functional elements
7 are precisely the ones that Google needed to use. RT 2159:23-2160:2 (Astrachan). Because the
8 declaration can only be written one way in the Java programming language—because the *idea*
9 underlying the declaration changes if the declaration is changed—any arguable expression in the
10 declaration merges with the underlying idea, and cannot be protected by copyright. *Herbert*
11 *Rosenthal Jewelry Corp. v. Kalpakian*, 446 F.2d 738, 742 (9th Cir. 1971); *Baker*, 101 U.S. at 104;
12 *Allen v. Academic Games League of Am., Inc.*, 89 F.3d 614, 617-18 (9th Cir. 1996). The
13 elements of the declarations are also functional requirements for compatibility with the J2SE API
14 packages, and thus cannot be protected by copyright for this separate reason. *Sega*, 977 F.2d at
15 1522 (citing 17 U.S.C. § 102(b)); *see also Lotus Dev. Corp. v. Borland Int’l, Inc.*, 49 F.3d 807,
16 815 (1st Cir. 1995), *aff’d by an equally divided court*, 516 U.S. 233 (1996).

17 **C. Question 3: The form of the fully qualified names of the methods in the 37**
18 **API packages is “package.class.method,” where the package names start with**
19 **“java.” or “javax.” and this form is required by the syntax of the Java**
language.

20 The fully-qualified name of a method has three parts: its package name, its class name
21 and its method name. RT 770:1-4 (Bloch). For example, the “cos” method (for calculating the
22 cosine of an angle) in the “Math” class in the “java.lang” package would be named
23 “java.lang.Math.cos”. RT 770:5-9 (Bloch). This format is required by the Java language. RT
24 770:10-12 (Bloch); TX 984 at 126-38.

25 **D. Question 4: Google could have come up with at least some different names**
26 **and SSO yet still provided similar functionality in Android, but this would**
not have been consistent with industry custom and developer demand.

27 It would have been possible in many instances for Google to have created APIs with
28 different names and/or SSO that would have provided similar functionality. That, however,

1 would have effectively undermined the ability to utilize the Java language, which language
2 provides a familiar environment for developers. By analogy, it is technically possible to create an
3 alternative version of the English language that follows the same rules of grammar, but in which
4 all the *words* are different. While such a scenario is possible, it would make little sense, and
5 would be contrary to the expectations of any reasonable person conversant in English. Moreover,
6 even if Google had done so, it still would have had to implement at least the parts of the 37
7 packages that are undisputedly *required* in order to implement the Java language, as explained
8 below.

9 **1. Many of the elements of the 37 API packages are *required* by the Java**
10 **language.**

11 In order to implement the Java language, Google was required to include many elements
12 of the 37 API packages. Dr. Reinhold admitted that at least 61 classes from the 37 API packages
13 are *required* by the Java language specification. RT 684:14-685:2; TX 1062 (Reinhold
14 summary); *see also* RT 1286:14-22 (Mitchell) (agreeing with Dr. Reinhold's analysis). These 61
15 classes are not merely mentioned by the language specification—"they are part of the
16 specification rather than being part of an example." RT 677:15-16 (Reinhold); *see also* RT
17 679:18-21. In addition, Oracle's Java language compiler depends on the presence of over 30
18 classes, including several that are not included in Dr. Reinhold's list of the 61 required classes in
19 TX 1062. *See* RT 679:22-681:21 (Reinhold); TX 1063 (Reinhold summary).

20 Although the Java language specification *requires* the presence of these classes, it does
21 not specify their details. RT 679:20-21 (Reinhold). Instead, those classes are specified in the *API*
22 specifications, which specify that those classes require over 750 methods and fields. RT 776:21-
23 777:9 (Bloch). Moreover, due to dependencies, implementing those 750 classes, method and
24 fields, requires implementation of 177 classes, with over 2,000 public methods and fields, spread
25 across ten of the accused API packages. RT 779:13-780:15 (Bloch).

26 Thus, *at least* portions of 10 of the 37 accused API packages must be implemented simply
27 to implement the Java language as required by the language specification, including both classes
28 required by the language specification, and those that the required classes depend upon.

1 **2. Industry custom and developer demand require implementation of the**
2 **37 API packages.**

3 Without the APIs, the Java language is a “primitive thing.” RT 686:15 (Reinhold).

4 Without APIs, the Java language can be used to “waste time,” but “that’s pretty much it.” RT
5 782:9-11 (Bloch); *see also* RT 782:12-783:18 (Bloch); RT 683:14-684:4 (Reinhold).

6 Developers expect the presence of the 37 API packages when they write programs in the
7 Java language. RT 2202:6-11 (Astrachan); RT 1782:6-1783:10 (Bornstein). Those packages are
8 needed in order to meet industry expectations. RT 2203:11-15 (Astrachan); RT 2291:1-8
9 (Mitchell); RT 519:21-23 (Screven). Indeed, programmers often memorize the names and
10 organization of members of these packages in order to help them write programs more efficiently.
11 RT 767:1-17 (Bloch); RT 2169:25-2170:13 (Astrachan); RT 2289:24-2290:3 (Mitchell). The 37
12 API packages are also necessary to make practical use of the language. RT 2196:7-2201:17
13 (Astrachan).

14 Sun recognized these facts, and indeed promoted widespread use of the Java language and
15 APIs. *See* RT 1957:24-1958:4, 1961:13-19, 1962:2-9 (Schwartz); RT 1474:24-1475:10, 1477:2-
16 1478:9 (Schmidt). Sun worked hard to dispel any suggestion that the SSO of the 37 API
17 packages was proprietary or protected. RT 1966:1-12 (Schwartz). Sun’s goal was to ensure that
18 the Java language was widely adopted by encouraging its teaching in colleges and universities.
19 RT 1476:9-14 (Schmidt); RT 1958:5-20 (Schwartz). Java language developers have always
20 understood that the Java API packages, along with the Java language, are free to use. RT 962:4-
21 14 (Swetland); RT 861:9-23 (Lindholm); RT 1769:18-1770:1 (Bornstein).

22 In view of this undisputed testimony, the APIs in the 37 packages are the Java language
23 equivalents of *scenes a faire*—and therefore uncopyrightable. *See Computer Assocs. Int’l, Inc. v.*
24 *Altai, Inc.*, 982 F.2d 693, 706-10 (2d Cir. 1992); *Swirsky v. Carey*, 376 F.3d 841, 850 (9th Cir.
25 2004); *Sega*, 977 F.2d at 1524 (constraints on the defendant are relevant to whether copyright
26 protection allowed); *Gates Rubber Co. v. Bando Chem. Indus., Ltd*, 9 F.3d 823, 838 (10th Cir.
27 1993).

1 **3. When developing new APIs in the Java language, it is standard**
 2 **practice to build upon the standard J2SE APIs.**

3 Dr. Reinhold testified that even when people design their own APIs, “[t]hey are building
 4 typically *on top of all of the standard Java APIs* and creating their own APIs for whatever
 5 problem they are trying to solve.” RT 685:12-14 (emphasis added). Thus, for example, when
 6 Wall Street firms create their own APIs for financial trading, “those are *strictly built on top of all*
 7 *of this Java platform stuff* we have been speaking about.” RT 685:18-20 (emphasis added).

8 **4. The Court should disregard Mr. Ellison’s testimony about Spring.**

9 When Larry Ellison was asked whether the Java APIs are needed to use the Java language,
 10 Google objected on the ground that the question called for expert testimony. RT 290:15-19. The
 11 Court overruled the objection, but only after Mr. Ellison assured the Court that he was testifying
 12 based on personal knowledge. RT 290:20-24. Mr. Ellison then testified that a UK company
 13 named Spring had built its “own Java environment” called Spring, which used the Java language,
 14 but not the Java APIs. RT 290:25-291:6.

15 Mr. Ellison’s testimony was incorrect. The Spring framework is open source software,
 16 and the documentation for the Spring framework is readily available on the Internet.⁴ This
 17 documentation demonstrates that Mr. Ellison’s testimony was incorrect, and that the Spring
 18 framework *uses the J2SE APIs*. For example, the Spring package “org.springframework.ui” has a
 19 class named “ModelMap,” which is a subclass of “java.util.HashMap”⁵—a class that is in the
 20 accused java.util package. This Spring class implements the interfaces Serializable (part of the
 21 accused java.io package), Cloneable (part of the accused java.lang package) and Map (part of the
 22 accused java.util package).⁶

23 This evidence, which flatly contradicts Mr. Ellison’s testimony, is not in the trial record.
 24 The accuracy of the cited documentation, however, cannot reasonably be questioned under the
 25 circumstances, and thus the Court may take judicial notice of these facts. Fed. R. Evid. 201(b)(2).
 26 Moreover, if the Court requests, Google will submit a declaration from Professor Astrachan

27 ⁴ See <http://static.springsource.org/spring/docs/2.0.x/api/index.html>.

28 ⁵ See <http://static.springsource.org/spring/docs/2.0.x/api/org/springframework/ui/ModelMap.html>.

⁶ See *id.*

1 explaining that the source code for the Spring framework depends on no less than 20 of the 37
 2 accused J2SE API packages. In light of these facts, Google requests that the Court not rely on
 3 any of Mr. Ellison's unsupported testimony about Spring. *See* RT 290:25-291:6, 304:13-22.

4 **E. Question 5: The input-output scheme of a method is not copyrightable.**

5 The input-output scheme of a method cannot be copyrighted because it represents an *idea*,
 6 not creative expression. 17 U.S.C. § 102(b). Moreover, the input-output scheme is required for
 7 compatibility with code that “calls” or “invokes” the method, which is another reason it cannot be
 8 protected by copyright. *See supra*, Part II.B; *Sega*, 977 F.2d at 1522.

9 In addition, there are only a limited number of ways in which to devise an input-output
 10 scheme for any given method. Independent of *Sega*, the merger doctrine bars copyright
 11 protection for input-output schemes. *See Allen*, 89 F.3d at 617-18; *Engineering Dynamics, Inc. v.*
 12 *Structural Software, Inc.*, 46 F.3d 408, 409-10 (5th Cir. 1995), *clarifying* 26 F.3d 1335 (5th Cir.
 13 1994); *see also* Google 4/3/12 Br. [Dkt. 852] at 8:19-9:20.

14 **F. Question 6: The “core” packages in 1996 included at least java.lang, java.io
 15 and java.util, and today include at least 34 of the accused packages.**

16 Google agrees that *at least* java.lang, java.io and java.util were “core” packages *in 1996*.
 17 Sun's documentation from that time stated that java.lang, java.io and java.util “must be included
 18 in all general purpose Java systems.” TX 2564 (*The Java Language Specification*, First Edition)
 19 at 31. This same book referred to those three packages as “core packages.” TX 2564 at 23.
 20 Another Sun book published in 1996, *The Java Application Programming Interface, Volume 1*,
 21 described those packages as “the foundation of the Java language” and as “general purpose
 22 libraries fundamental to every Java program.” TX 980 at 528. This book defined the “core
 23 packages” to be those three packages, plus java.net. TX 980 at xix, back cover. As the Court has
 24 recognized, the case against copyright protection and for fair use for the “core” packages is even
 25 stronger than it is for the rest of the accused packages. RT 3388:12-3389:6, 3389:21-3390:11.

26 Later versions of J2SE, however, recognize many more packages as “core” packages. For
 27 example, the documentation for J2SE 1.4 defines the “core” packages to include *all but four of*
 28 *the accused packages*—every accused package except java.lang.annotation, java.security.acl,

1 java.sql and javax.sql. See TX 622 (source code for J2SE 1.4, which also includes
2 documentation).⁷

3 The first of the four remaining API packages, java.lang.annotation, was not introduced
4 until J2SE 5.0. It is one of the few packages that is *directly required* by the third edition of *The*
5 *Java Language Specification*, which includes an entire section about “Annotations.” TX 984 at
6 270-86. This section states that “[t]he direct superinterface of any annotation type is always
7 [java.lang.]⁸ annotation.Annotation”—i.e., the Annotation interface in the java.lang.annotation
8 package. *Id.* at 272. The section also discusses four predefined annotation types—Target,
9 Retention, Inherited and Override—that are part of the java.lang.annotation package. *Id.* at 277-
10 79. For J2SE 5.0, java.lang.annotation is unquestionably a core package.

11 This leaves three accused packages, java.security.acl, java.sql and javax.sql. The first of
12 these, java.security.acl, appears to have been left out of the list of “core” packages in J2SE 1.4
13 only by accident. Each of the other subpackages of java.security (java.security.cert,
14 java.security.interfaces and java.security.spec) is listed. See TX 622.⁹

15 The final two packages, java.sql and javax.sql, are not identified as “core” packages in the
16 J2SE 1.4 documentation. However, these two packages cover such basic concepts for data
17 storage and retrieval that they should be considered, as a practical matter, to be required elements
18 of the Java language. RT 2198:25-2199:16 (Astrachan).

19 **G. Question 7: The Java language requires more than just the java.lang, java.io**
20 **and java.util packages.**

21 The Java language requires methods, classes and packages beyond java.lang, java.io and
22 java.util. This is because the 61 classes that are *directly* required in order to implement the Java
23 language themselves are dependent on other classes. The full chain of interdependencies requires

24 ⁷ The relevant documentation can also be accessed on the web at
25 <http://docs.oracle.com/javase/1.4.2/docs/guide/core/index.html>. Following the links on that page
leads to webpages that include lists of API packages. There does not appear to be an analogous
list present in the J2SE 5.0 documentation.

26 ⁸ In *The Java Language Specification*, where no package is expressly identified, “the intended
27 reference is to the class or interface . . . in the package java.lang.” TX 984 at 6.

28 ⁹ For the Court’s convenience, the specific page is also available on the web at
<http://docs.oracle.com/javase/1.4.2/docs/guide/security/index.html>.

over 177 classes, with over 2,000 public methods and fields, drawn from ten of the accused API packages. *See supra*, Part II.D.1.

H. Question 8: All 37 accused API packages should be deemed “core” packages.

Based on the documentation for J2SE 1.4, and the third edition of *The Java Language Specification*, 34 of the accused API packages are concededly core packages. One of the remaining three, `java.security.acl`, appears to have been left off of the list of core packages in J2SE 1.4 only by mistake. The final two packages, `java.sql` and `javax.sql`, cover such fundamental concepts for modern software applications that they, too, should be deemed core packages. *See supra*, Part II.F.

I. Question 9: There are cross-method, cross-class interdependencies at the implementation level in J2SE, and those implementation level interdependencies are not always duplicated in the Android implementations.

The Court asked the parties to address, in their April 22, 2012 briefs, whether “any of the Sun compiled lines in the 37 APIs call upon part or all of another API as a step” and, if so, whether Android’s implementing code “likewise call upon the same other API.” *See* Order re Brief Due Sunday [Dkt. 951] at 1. Google’s brief responded that, yes, the J2SE implementations do reference other methods and classes, but, no, the Android implementations do not necessarily follow the same pattern. *See* Dkt. 955 at 13:8-14:6. Oracle’s brief addressed references in class, method, and field *declarations* (which are necessarily similar for compatibility reasons), rather than references in the *implementations* (which can be different while remaining compatible). *See* Dkt. 956 at 13:15-14:10.

At trial, there was no testimony on these points. The source code that is in evidence in native format, however, confirms that the positions Google advanced in its April 22, 2012 brief—positions Oracle has not denied—are correct. *See* TX 623 (J2SE 5.0 source code), TX 46 (Android, “Froyo” release, source code). If the Court requests, Google will submit a declaration attaching printed excerpts from those trial exhibits demonstrating that, for example, Android’s implementation of the `URL` class in the `java.net` package makes use of the `ObjectOutputStream` class from the `java.io` package, while the J2SE implementation makes use of the `OutputStream` class from the `java.io` package.

J. Question 10: At the name/declaration level, the only interdependencies in the Java language come from (1) the package/class/member organizational scheme; (2) inheritance via subclassing and subinterfacing; and (3) interface implementation.

There are three types of interdependencies at the name/declaration level. First, there is the organizational scheme that groups class members (such as methods) into classes, and classes into packages. Second, classes and interfaces can inherit characteristics from their superclasses and superinterfaces. Third, classes can implement interfaces. RT 2187:18-2188:24 (Astrachan); RT 584:8-603:6 (Reinhold); TX 1028 (key, showing only class/package, interface, and subclass/superclass relationships). These are the only name/declaration level interdependencies that were identified at trial.

K. Question 11: The Seventh Circuit's decision in *American Dental Ass'n v. Delta Dental Plans Ass'n*, was either wrongly decided, or unclear about the scope of its holding.

The Seventh Circuit's *American Dental Ass'n* decision purported to address whether the ADA's "taxonomy" for dental procedures could be copyrighted. 126 F.3d 977, 977 (7th Cir. 1997). But in concluding that the "taxonomy" was expressive, the Seventh Circuit relied on the *text descriptions* the ADA employed. *See id.* at 979. That suggests that the court's decision was about the copyrightability of the ADA's *book describing the taxonomy*, not the taxonomy separate and apart from those descriptions. If so, the decision is not relevant to the copyrightability determination that the Court must make in this case. If, however, the Seventh Circuit did intend to hold that the taxonomy itself was copyrightable, its reliance on text descriptions of the taxonomy to conclude that the taxonomy itself was copyrightable was nonsensical, and renders the decision of suspect persuasive value.

No Ninth Circuit case has ever cited *American Dental Ass'n*. Indeed, the Seventh Circuit's decision has only been cited three times by *any* court. The Seventh Circuit itself has cited the *American Dental Ass'n* decision only once, and only for propositions unrelated to the "taxonomy" holding. *Seng-Tiong Ho v. Taflove*, 648 F.3d 489, 497-500 (7th Cir. 2011). The Third Circuit has cited the decision once, but found the facts of the case before it factually distinguishable. *See Southco, Inc. v. Kanebridge Corp.*, 258 F.3d 148, 155 (3d Cir. 2001).

1 Finally, the Sixth Circuit *criticized* the decision, calling the Seventh Circuit’s reasoning “opaque.”
 2 *ATC Distrib. Grp., Inc. v. Whatever It Takes Trans. & Parts, Inc.*, 402 F.3d 700, 708 (6th Cir.
 3 2005).

4 Even if the Court were to assume that a taxonomy can be copyrighted—and to do so
 5 would be legal error, because the ADA taxonomy is an unprotectable system, *see* 17 U.S.C.
 6 § 102(b)—that does not mean that the SSO of the 37 API packages should be protected by
 7 copyright. The Seventh Circuit rejected the argument that the “taxonomy” was an unprotectable
 8 system, noting that “[t]his taxonomy does not come with instructions for use, as if the Code were
 9 a recipe for a new dish.” 126 F.3d at 980. The 37 API packages, however, *do* come with
 10 instructions for use—that is precisely what the API specifications are. The APIs in the 37
 11 accused packages are a system by which Java language developers express themselves, and as
 12 such are an uncopyrightable system. 17 U.S.C. § 102(b).

13 Finally, although the Seventh Circuit *purported* to hold that a “taxonomy” can be
 14 copyrighted, nothing in the decision supports the conclusion that a numbering system can be
 15 copyrightable separate and apart from descriptions of the parts of the system. In *American Dental*
 16 *Ass’n*, the Seventh Circuit ultimately held only that the defendant could not copy “the Code
 17 itself” or distribute derivative works based on “the Code.” 126 F.3d at 981. But the opinion
 18 earlier defined “the Code” to be *a book*. *See id.* at 977 (“The American Dental Association has
 19 created the *Code on Dental Procedures and Nomenclature*. The first edition was published in
 20 1969; the Code has been revised frequently since, in response to changes in dental knowledge and
 21 technology.”). The court also referred to “the numbering system and short descriptions from the
 22 ADA’s Code.” *Id.* If the phrase “the Code” referred to the numbering system itself, separate
 23 from the descriptions, then the reference to “the numbering system . . . from the ADA’s Code”
 24 would mean “the numbering system . . . from the ADA’s numbering system,” which is
 25 recursively meaningless, referring to “the numbering system” as something that can be extracted
 26 from “the numbering system.”

27 Thus, the better reading of the decision is that “the Code” refers to *the ADA’s book*
 28 *describing the numbering system*, complete with “the short descriptions” of each procedure. *See*

1 *id.* So understood, the Seventh Circuit’s holding that “the Code” is copyrightable, *see id.* at 979,
 2 means only that the ADA’s *book describing its numbering system* is copyrightable. If that is the
 3 extent of the Seventh Circuit’s holding, then the decision is irrelevant to the present case. If the
 4 Seventh Circuit instead intended to hold that the *numbering system*, separate from its descriptions,
 5 was protected by copyright, the decision is unclear, poorly reasoned, and should not be followed
 6 absent Ninth Circuit precedent for doing so—and there is none.

7 **L. Question 12: *CDN, Inc. v. Kapes* was not about a method of operation or**
 8 **system.**

9 In *CDN, Inc. v. Kapes*, the Ninth Circuit did not address whether the SSO of CDN’s price
 10 list was protectable, because there was no allegation that the SSO had been copied. 197 F.3d
 11 1256, 1259 (9th Cir. 1999) (“CDN does not allege that Kapes copied the entire lists, as the
 12 alleged infringer had in *Feist*. . . . Thus Kapes’ argument that the selection is obvious or dictated
 13 by industry standards is irrelevant.”). The case therefore sheds no light on whether the SSO of
 14 the 37 API packages is copyrightable.

15 Instead of SSO, the issue in *CDN* was whether CDN’s estimates of coin prices were
 16 “sufficiently original *as compilations* to sustain a copyright.” *Id.* (emphasis added). The Ninth
 17 Circuit held that CDN relied on pricing information provided by others (i.e., that the coin prices
 18 in its guide were not of its own creation). *Id.* at 1260. CDN then *chose* which coin prices to *keep*,
 19 “retain[ing] only that information it considers to be the most accurate and important.” *Id.* “The
 20 prices CDN creates are compilations of data that represent its best estimate of the value of the
 21 coins.” *Id.*

22 This theory of copyrightability is not available to Oracle. In its April 3, 2012 brief, Oracle
 23 denied that its APIs were a compilation. *See* Oracle 4/3/12 Br. [Dkt. 853] at 1:7-8 (“The 37 APIs
 24 should not be viewed as a compilation under section 101 of the Copyright Act.”). After briefly
 25 attempting to switch course, Oracle again conceded that it was not arguing for protection under a
 26 collective work theory. *See* RT 2134:11-17. At no time since has Oracle argued that the
 27 elements of the 37 API packages should be protected under a compilation theory.

28 Moreover, *CDN* did not address section 102(b)’s system and method of operation

1 exclusions, which are central to the Court’s copyrightability determination in this case. The
 2 Ninth Circuit opened its decision by stating, “We must decide whether prices listed in a wholesale
 3 coin price guide contain *sufficient originality* to merit the protection of the copyright laws.” 197
 4 F.3d at 1257 (emphasis added). However, Kapes also argued that CDN’s prices were
 5 unprotectable *ideas*. *See id.* at 1261. The Ninth Circuit disagreed, holding that CDN’s “prices
 6 fall on the expression side” of the idea/expression dichotomy. *Id.* at 1262.

7 CDN, however, did not seek copyright protection for a *system* of coin pricing, or a *method*
 8 *of operation* for coin pricing. Had it done so, its argument would have failed, because CDN
 9 could not “claim protection for its idea of creating a wholesale price guide” *Id.*; *see also* 17
 10 U.S.C. § 102(b). Moreover, Kapes did not argue that functional requirements for compatibility
 11 compelled him to use the same coin prices that CDN did. *See Sega*, 977 F.2d at 1522.

12 **M. Question 13: Android is compatible with the 37 API packages at issue**
 13 **because code written to use those APIs will compile and work properly on an**
 14 **Android device.**

14 The Court has asked the parties to address what they mean when they refer to
 15 “compatibility.” Android is compatible with the APIs from the 37 API packages and, as a
 16 necessary part of that compatibility, it has substantially the same SSO as the 37 API packages.

17 Compatibility is not an all or nothing proposition. Two things can be compatible in some
 18 respects, but not others. The key issue in the present case is not whether Android is fully
 19 compatible with J2SE in all respects, but whether it is compatible with the APIs in the 37 API
 20 packages in the computer science sense—not “compatible” in Sun’s or Oracle’s business plan
 21 sense. The parties’ experts agreed that the platforms are compatible from the perspective of
 22 computer science, because code written using the APIs in those packages will work on both
 23 platforms. RT 2171:24-2172:11 (Astrachan); RT 2292:25-2293:14 (Mitchell). For example,
 24 Professor Astrachan wrote the following program during trial:

```

25 package simple;
26 /** @author ola */
27 public class WebReader {
28     public static void main (String[] args) {
29         java.net.URL site=new java.net.URL("http://cnn.com");
30         java.io.InputStream source=site.openStream();
  
```

```

1      System.out.print(source.read());
2      }
3  }

```

TX 3536 (demonstrative); RT 2162:12-2167:25 (Astrachan). This code defines a package named “simple,” which includes a class named “WebReader.” RT 2163:5-22. Within that class is a short method that (1) creates a “URL object” associated with the URL <http://cnn.com>; (2) creates an “InputStream” object by opening a stream to the CNN website; and (3) displays the first character from that website. RT 2165:17-2167:25, 2170:22-2171:3. This program depends upon elements and invokes methods from three of the accused API packages, java.net, java.io and java.lang. RT 2169:20-24. Because the methods that the program invokes and the API elements it relies on are implemented both in J2SE and Android, this program is compatible with both of those platforms. RT 2171:19-23; *see also* RT 2292:25-2293:8 (Mitchell) (the three lines of code Professor Astrachan wrote that use the J2SE APIs would work on both the J2SE and Android platforms). This understanding of “compatibility” is not a position adopted just for this litigation. Indeed, Oracle’s expert called this definition of compatibility “a great definition of ‘compatible’” RT 2293:9-14 (Mitchell).

16 **N. Questions 14-15: In the Java language, “inheritance” is a concept applicable**
17 **to classes, not packages; by virtue of the Java language specification, a**
18 **subclass inherits fields, methods, and other members from its superclass.**

In the Java language, if “you define one class to be a subclass of another, then the subclass inherits all the methods of the superclass” RT 1225:13-15 (Mitchell); *see also* RT 2188:10-11 (Astrachan). The inheritance relationship exists at the class level. RT 2243:6-7 (Astrachan) (“the class declaration, which shows the inheritance relationships and the interface relationships”). There is no such thing as “inheritance” of a package. *See* TX 984 at 617 (index; no discussion of “package” under “inheritance”); *see also id.* at 154 (recognizing that hierarchical naming structure for packages “has no significance in itself” and that similarly-named packages have no “special access relationship “to each other”).

Inheritance is a characteristic of a class or interface that results from the superclass/subclass and superinterface/subinterface relationships that are part of the Java language. *See* TX 984 at 188, 190. A class or interface inherits “public” members (fields,

1 methods, etc.) of its superclass or superinterface, as well as any members declared in the class or
 2 interface itself. *See id.*

3 **O. Question 16: Copyright protection does not extend to names, including fully**
 4 **qualified names, nor does it extend to input-output (argument-return)**
 5 **designations, exception types, subclass inheritances, or interface relationships.**

6 The Court has asked what Google allegedly copied aside from names (including fully-
 7 qualified names) and input-output (argument-return) designations. In addition to those items, the
 8 SSO of the Android APIs for the 37 packages at issue adopts substantially the same “exceptions”
 9 for methods (i.e., the methods “throw” or generate the same error messages), subclass and
 10 subinterface structures for classes and interfaces (i.e., inheritance) and interface implementations
 11 for classes. However, the exception, inheritance and interface relationships are all part of the
 12 “ideas” underlying the defined methods, classes and interfaces. These relationships are therefore
 13 uncopyrightable by virtue of the merger, *scenes a faire* and/or functional requirements for
 14 compatibility doctrines, and/or the system and method of operation exclusions of section 102(b)
 15 of the Copyright Act.

16 **III. The Court should hold that the SSO of the 37 API packages is not copyrightable.**

17 The parties have submitted voluminous briefing on this legal issue, and much of the
 18 evidence during Phase One of the trial related to the issue of copyrightability. There are many
 19 reasons why the Court should conclude that the SSO of the 37 API packages is not copyrightable,
 20 but the Court has no doubt read the prior briefs carefully, and Google will not repeat each of its
 21 arguments here once more.

22 The simple point, however, is that Oracle seeks to misuse copyright to control a
 23 language—to restrict dramatically the right to use the established vocabulary of a freely-available
 24 language and to require that others who want to use the language create an entirely new
 25 vocabulary. Whether the APIs at issue are part of the “Java language” is important to deciding
 26 the effect of Oracle’s concession that it does not assert copyright protection in this case over the
 27 Java language itself. Regardless of how the Court resolves that point, however, the APIs are part
 28 of a language. There is no dispute that when Java developers use the APIs, they do so to express
 themselves. Indeed, without APIs, the Java language is, in all practical effect, useless. RT

1 683:14-684:4, 707:18-21 (Reinhold); RT 782:9-14 (Bloch); RT 1477:2-13 (Schmidt); RT 1960:4-
2 8 (Schwartz).

3 Oracle argues that developers can create *new* APIs, and then use those new APIs *instead*
4 of the J2SE APIs, and thus can still express themselves in the Java language without using the
5 J2SE APIs. That is, at least to some extent, wrong, because the Java language cannot be
6 implemented without at least *some* of the APIs at issue. RT 1274:16-24 (Mitchell); RT 684:16-
7 685:2, 679:12-21 (Reinhold); RT 776:12-777:9, 777:19-778:9, 779:13-780:18 (Bloch). But it is
8 also irrelevant. The language that developers use when they write programs in the Java language
9 with the J2SE APIs—whether that language is called the “Java language” or “the Java language
10 with the J2SE APIs”—is a functional system that can be used for expression and is known to
11 millions of developers. That system cannot be protected by copyright. 17 U.S.C. § 102(b).

12 Allowing Oracle to use copyright law to control the J2SE APIs would in effect grant
13 Oracle a monopoly, allowing it to prevent millions of Java developers from using their skills to
14 write applications for platforms that Oracle has not approved. This goes beyond protecting
15 *Oracle’s* expression, and would grant Oracle control over how *developers* are allowed to express
16 themselves.

17 It is no answer to say that there are *other* systems that developers can use to express
18 themselves. Of course that is true—we know it is true, because there are many different
19 programming languages that are the result of many different design choices made by language
20 designers. But section 102(b) does not exclude systems from copyright protection only when
21 there are no alternatives. It states that “[i]n no case” shall copyright protection extend to a
22 system. 17 U.S.C. § 102(b).

23 And there is a strong policy reason for that stark prohibition: choosing one system over
24 another is not a matter of expression, it is a matter of choosing one idea over another. The J2SE
25 APIs include a “collections framework” that was designed by Josh Bloch at least in part when he
26 was a Sun employee. RT 750:5-7, 750:16-751:4 (Bloch). The approach Dr. Bloch adopted for
27 the collections framework was, in at least one person’s estimation, life changing. RT 750:8-12
28 (Bloch). The notion of “collections” was not new—other, older languages implemented

1 collections frameworks of their own, which conceptualized collections in different ways. RT
 2 1242:16-1243:3 (Mitchell). The reason Dr. Bloch's collections framework was life-changing was
 3 that, at least for that one developer, Dr. Bloch's idea was better than those that had come before.
 4 His collections framework was a better mousetrap—a better idea.

5 Because copyright does not grant a monopoly over ideas, when someone comes up with a
 6 good idea, others are free to adopt it absent some other legal principle—such as patent law—
 7 protecting the idea. Good ideas rarely come easily. Indeed, they may take years of effort and
 8 experimentation. They are still, however, ideas, and still unprotected by copyright law.

9 The record shows that the SSO of the 37 packages—to the extent it includes anything
 10 other than the “structure” imposed by the Java language hierarchical naming rules—is a method
 11 of operation, a system of communicating with software, and functionally required for
 12 compatibility. The record also shows that the form of expression in the source code that reflects
 13 the SSO is constrained by the requirements of the language and therefore unprotectable under the
 14 merger doctrine. Finally, the record shows that any arguable expression in the SSO has come to
 15 be expected by programmers and industry, is required for compatibility purposes and reflects
 16 widely-accepted programming practices—and cannot therefore form the basis for an infringement
 17 action. For all the reasons given in its prior briefs, as well as the reasons given above, Google
 18 respectfully requests that the Court hold that the SSO of the 37 API packages at issue is not
 19 copyrightable.

20 Dated: May 10, 2012

KEKER & VAN NEST LLP

21
 22 By: /s/ Robert A. Van Nest
 ROBERT A. VAN NEST

23 Attorneys for Defendant
 24 GOOGLE INC.